Theoretische Informatik - Übungsblatt 10

Jan Hendrik Dithmar - Matrikelnummer 2031259
Dirk Heine - Matrikelnummer 2030941
Di, 9-11 Uhr, Andreas Augustin

Aufgabe 1

bearbeitet von Dirk Heine
Laut Definition gilt:

$$A \leq B \Rightarrow \exists f : \Sigma^* \to \Sigma^* : w \in A \Leftrightarrow f(w) \in B$$

$$B \leq C \Rightarrow \exists g : \Sigma^* \to \Sigma^* : v \in B \Leftrightarrow g(v) \in C$$

f und g sind total rekursiv und überall definiert.

Idee: Wendet man auf eine Eingabe w zunächst die Funktion g, und auf das Ergebnis dann die Funktion f an, so ist die Transitivität nach Definition bereits gezeigt.

$$A \leq C \Rightarrow \exists h : \Sigma^* \to \Sigma^* : w \in A \Leftrightarrow h(w) \in C \text{ laut Definition}$$

Sieht man nun die Funktion h als Konkatenation der beiden Funktionen f und g an, so gilt:

$$A \leq B \leq C \Rightarrow \exists f, g : \Sigma^* \to \Sigma^* : w \in A \Leftrightarrow g(w) \in B \Leftrightarrow f(g(w)) \in C$$

Also gilt insbesondere auch:

$$A \leq C \Rightarrow \exists f, g : \Sigma^* \to \Sigma^* : w \in A \Leftrightarrow f(g(w)) \in C$$

Konstruktion:
Sei f die zur Reduktion $A \leq B$, und g die zur Reduktion $B \leq C$ gehörende Funktion. Sei M_2 eine (stets haltende) Turing-Maschine, die genau die Wörter aus C akzeptiert. Dann bauen wir eine (stets haltende) Turing-Maschine M_1, die genau die Wörter aus A akzeptiert, wie folgt: M_1 wendet zunächst die Funktion g auf die Eingabe w an, und berechnet $g(w)$. Dann wendet M_1 auf $g(w)$ die Funktion f an, und berechnet $f(g(w))$. Dann simuliert M_1 auf $f(g(w))$ die Maschine M_2 und akzeptiert w genau dann, wenn M_2 das Wort $f(g(w))$ akzeptiert. Laut Definition ist dies korrekt.

Aufgabe 2

bearbeitet von Jan Hendrik Dithmar
Quelle: „Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie“, Hopcroft, 2. Auflage

Behauptung:
Ist L_a eine kontextfreie Sprache für die Liste A, dann ist $\overline{L}_a$ eine kontextfreie Sprache.

Beweis:
Sei Σ das Alphabet der Zeichenreihe aus Liste $A = w_1 w_2 ... w_k$ und $I = \{a_1, ..., a_k\}$ die Menge der Indexsymbole. Wir entwerfen einen DKA P, der $\overline{L}_a$ akzeptiert und wie folgt arbeitet:

1. Solange P Symbole aus Σ sieht, werden diese auf dem Stack gespeichert. Da alle Zeichenreihen aus Σ^* in $\overline{L}_a$ enthalten sind, werden sie im weiteren Verlauf von P akzeptiert.

2. Sobald P ein Indexsymbol aus I sieht, etwa a_i, werdem Symbole vom Stack entfernt und P prüft, ob die obersten Symbole w_i^R bilden. w_i^R stellt dabei die Spiegelung der korrespondierenden Zeichenreihe dar.

   a) Falls nicht, dann ist die bisherige Eingabe sowie jede Fortsetzung dieser Eingabe in $\overline{L}_a$ enthalten. P wechselt daher in einen akzeptierenden Zustand und verarbeitet alle weiteren Eingaben, ohne dabei den Stack zu verändern.

   b) Wurde w_i^R vom Stack entfernt ohne dass die Stackanfangsmarkierung auf dem Stack sichtbar ist, dann akzeptiert P, erinnert sich in diesem Zustand jedoch daran, dass nur nach Symbolen aus I gesucht wird und eine Zeichenreihe aus L_a auftreten könnte, die P nicht akzeptiert. P wiederholt solange Schritt 2., wie die Frage nicht gelöst ist, ob die Eingabe in L_a enthalten ist.

   c) Wurde w_i^R vom Stack entfernt und ist die Stackanfangsmarkierung auf dem Stack sichtbar, dann hat P eine in L_a enthaltene Eingabe gesehen und akzeptiert diese Eingabe nicht. Folgen allerdings noch weitere Eingaben, wechselt P in einen Zustand, in dem alle weiteren Eingaben akzeptiert werden und der Stack unverändert bleibt.

3. Sieht P nach einem oder mehreren Symbolen aus I ein weiteres Symbol aus Σ, dann besitzt die Eingabe nicht die korrekte Form einer Zeichenreihe aus L_a. P wechselt daher in einen Zustand, in dem er diese sowie alle weiteren Eingaben akzeptiert ohne den Stack zu verändern.

Aufgabe 3

bearbeitet von Jan Hendrik Dithmar

(a)

Mit dieser Einschränkung ist das PCP entscheidbar, da das Problem auf die Länge der Strings reduziert wird, wenn das Alphabet nur ein Zeichen enthält.
Idee:
Es wird dabei angenommen, dass kein Teil enthalten ist, bei dem $|x_i| = |y_i|$, da sonst das Problem trivial lösbar wäre.

1. Finde (x_i, y_i) und (x_j, y_j) mit $(|x_i| > |x_j| \wedge |y_i| < |y_j|) \vee (|x_i| < |x_j| \wedge |y_i| > |y_j|)$. In diesem Fall besitzt das PCP keine Lösung, da die Strings offensichtlich nie gleich lang sein werden können, da mit jeder „Spielkarte“ der obere oder der untere String immer um mehr erhöht wird als der andere. $\longrightarrow$ ablehnen und terminieren.

2. Wurde das Paar gefunden, so ist $I = (\underbrace{i}_{||x_j|-|y_j||-mal}, \underbrace{j}_{||x_i|-|y_i||-mal})$ Lösung des PCP.

Beweis: Zu zeigen:

$$\begin{aligned} |x_i| \cdot ||x_j| - |y_j|| + |x_j| \cdot ||x_i| - |y_i|| &= |y_i| \cdot ||x_j| - |y_j|| + |y_j| \cdot ||x_i| - |y_i|| \\ \Leftrightarrow |x_i| \cdot ||x_j| - |y_j|| - |y_i| \cdot ||x_j| - |y_j|| &= |y_j| \cdot ||x_i| - |y_i|| - |x_j| \cdot ||x_i| - |y_i|| \\ \Leftrightarrow (|x_i| - |y_i|) \cdot (||x_j| - |y_j||) &= (|y_j| - |x_j|) \cdot (||x_j| - |y_j||) \end{aligned}$$

Durch eine Fallunterscheidung, ob nur $(|x_i| > |x_j| \wedge |y_i| < |y_j|)$ oder $(|x_i| < |x_j| \wedge |y_i| > |y_j|)$ für das gefundene Paar galt, lassen sich die Betragsstriche entfernen. Für beide Fälle ergibt die Gleichung offensichtlich wahre Aussagen.

1. Fall:

$$\begin{aligned} (|x_i| - |y_i|) \cdot (-|x_j| + |y_j|) &= (|y_j| - |x_j|) \cdot (|x_i| - |y_i|) \\ \Leftrightarrow (|x_i| - |y_i|) \cdot (-1) \cdot (|x_j| - |y_j|) &= (-1) \cdot (|x_j| - |y_j|) \cdot (|x_i| - |y_i|) \\ \Leftrightarrow -1 &= -1 \quad \text{(w)} \end{aligned}$$

2. Fall:

$$\begin{aligned} (|x_i| - |y_i|) \cdot (|x_j| - |y_j|) &= (|y_j| - |x_j|) \cdot (-|x_i| + |y_i|) \\ \Leftrightarrow (|x_i| - |y_i|) \cdot (-1) \cdot (|y_j| - |x_j|) &= (|y_j| - |x_j|) \cdot (-1) \cdot (|x_i| - |y_i|) \\ \Leftrightarrow (-1) &= (-1) \quad \text{(w)} \end{aligned}$$

(c)

Nicht entscheidbar. Beweis durch Gegenbeispiel:

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}; |\Sigma| = 8$$

Spielkarten:

$$(01, 0); (10, 1); (23, 2); (32, 3); (45, 4); (54, 5); (67, 6); (76, 7)$$

Fängt man z.B. mit $(01, 0)$ an, so muss man als nächstes zweimal (10,1) nehmen. Danach muss man wieder $(01, 0)$ nehmen; das Spiel geht von vorne los. Die obere Zeile der „Spielkarte“ bekommt dadurch einen enormen „Vorsprung“, der nicht wieder eingeholt werden kann. Somit läuft die TM ewig und das Problem ist nicht entscheidbar.